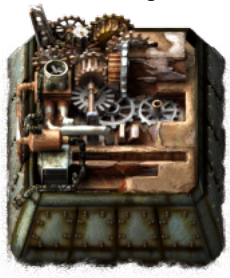
2. Cole

Program Name: Cole.java **Input File: cole.dat**

Cole is revisiting his old favorite game, *Factorio*, now with a new DLC. A core mechanic of the game is automation — namely, anything that can be crafted can also be automated using *Assembling Machines*. These machines take inputs, process recipes, and produce outputs, which can then be used in more advanced recipes.



Previously, Cole relied on online guides to determine the correct input-output ratios. Now, he's much smarter, and wants to calculate them himself. Given a set of crafting recipes and a target production rate for a specific item, determine how many assembling machines are needed for each input.

Input: The first line will consist of two space-separated integers R and Q, ($1 \leq R \leq 300$, $1 \leq Q \leq 10^4$), denoting the number of crafting recipes and queries. The next R lines will all take the same format:

$$S, p/t, n, (i_1, q_1), (i_2, q_2), \ldots, (i_n, q_n)$$

Where…

- S is the item name.
- p is the number of items produced per crafting cycle.
- t is the time taken per crafting cycle.
- n is the number of inputs required.
- Each pair (i_k, q_k) represents an input item and its required quantity.

Raw ingredients (available infinitely, but take time to harvest) have $n = 0$, $p = 1$, and $t = 1$, will be written as:

$$S, 1/1, 0$$

Every input item in a recipe is guaranteed to have a corresponding entry among the R recipes. Moreover, there are no cycles in the dependencies, and there will be *at least* one crafting recipe that is a raw ingredient.

The next Q lines contain queries in the format "A, r" where A is the item to be produced, and r ($10^{-4} < r \leq 10^6$) is the desired output rate per second (up to 5 decimal places). A is guaranteed to be in the recipes.

Output: For each of Cole's Q queries…

1. Print "A (r):", where r is formatted to 5 decimal places.
2. For each required input (i_k), print a tab followed by "i_k: a_k", where a_k is the number of assembling machines needed, rounded to the nearest whole number. List inputs in lexicographical order.

Note that we are assuming that r is an average rate that needs to be met over an effectively infinite amount of time and need not be satisfied at rate r immediately.

Hint: To avoid floating-point precision errors when rounding, subtract a small value $\varepsilon = 10^{-6}$ before rounding.

~ ***Sample input and output on next page*** ~

~ *Cole continued* ~

Sample input: (*indented lines are a continuation of the previous line*)

```
10 3
ExpressTransportBelt, 1/0.5, 3, (FastTransportBelt, 1), (IronGearWheel, 10),
      (Lubricant, 20)
FastTransportBelt, 1/0.5, 2, (TransportBelt, 1), (IronGearWheel, 5)
TransportBelt, 2/0.5, 2, (IronGearWheel, 1), (IronPlate, 1)
IronGearWheel, 1/0.5, 1, (IronPlate, 2)
IronPlate, 1/3.2, 1, (IronOre, 1)
Lubricant, 10/1, 1, (HeavyOil, 10)
HeavyOil, 25/5, 2, (CrudeOil, 100), (Water, 50)
IronOre, 1/1, 0
CrudeOil, 1/1, 0
Water, 1/1, 0
TransportBelt, 20
FastTransportBelt, 10
ExpressTransportBelt, 5
```

Sample output:

```
TransportBelt (20.00000/s):
      IronGearWheel: 5
      IronOre: 30
      IronPlate: 96
      TransportBelt: 5
FastTransportBelt (10.00000/s):
      FastTransportBelt: 5
      IronGearWheel: 28
      IronOre: 115
      IronPlate: 368
      TransportBelt: 3
ExpressTransportBelt (5.00000/s):
      CrudeOil: 400
      ExpressTransportBelt: 3
      FastTransportBelt: 3
      HeavyOil: 20
      IronGearWheel: 39
      IronOre: 158
      IronPlate: 504
      Lubricant: 10
      TransportBelt: 2
      Water: 200
```